Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ_____

КАФЕДРА _____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Анализ рекомендательных систем (content-base, collaboration filtering, hybrid filtering)

Студент _ИУ5-33М_ _____ **_____Семенов ИА.____**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **_____Гапанюк Ю.Е.__**
(Подпись, дата) (И.О.Фамилия)

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой __ИУ-5__
(Индекс)
_____ В.И.Терехов
(И.О.Фамилия)
« _____ » _____ 2024 г.

ЗАДАНИЕ
на выполнение научно-исследовательской работы

по теме __Анализ рекомендательных систем (content-base, collaboration filtering, hybrid filtering)_____

Студент группы __ИУ5-23М

_____Семенов Илья Александрович_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

___учебная_____

Источник тематики (кафедра, предприятие, НИР) ___учебная тематика_____

График выполнения НИР: 25% к __12__ нед., 50% к __14__ нед., 75% к __15__ нед., 100% к __16__ нед.

Техническое задание *_Обзор основных методов анализа текстовых данных, разработка системы для автоматизированного анализа текстовых данных из социальных медиа с целью выявления трендов и настроений.*_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 11 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_Слайды презентации 7-8 шт._____
_____
__

Дата выдачи задания « 14 » _сентября_ 2024 г.

Руководитель НИР _____ _____ Гапанюк Ю.Е.
(Подпись, дата И.О.Фамилия)

Студентк _____ __.__.2024 г. Семенов И.А.
(Подпись, дата И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Введение

Рекомендательные системы являются неотъемлемой частью современной цифровой среды, играя ключевую роль в адаптации контента под индивидуальные предпочтения пользователей. Они применяются в самых разных областях: от интернет-магазинов до стриминговых платформ, где помогают пользователям находить фильмы, музыку, товары и другие материалы, соответствующие их интересам. Основные подходы к разработке таких систем включают content-based filtering (контентно-ориентированная фильтрация), collaborative filtering (коллаборативная фильтрация) и их комбинации в виде гибридных методов.

Проблема заключается в сложности выбора рекомендательной системы с точки зрения качества рекомендаций, времени генераций рекомендации и используемой памяти для генерации рекомендаций.

Несмотря на значительный прогресс в разработке алгоритмов и повышении точности рекомендаций, эффективность и удобство работы с рекомендательными системами зависят не только от технических характеристик, но и от эргономики. Эргономический анализ таких систем позволяет оценить, насколько они отвечают требованиям пользователей с точки зрения быстродействия, точности, прозрачности, удобства использования и адаптивности. Это особенно важно для создания интуитивно понятных и эффективных решений, которые удовлетворяют ожидания как рядовых пользователей, так и профессионалов.

Данная работа посвящена исследованию эргономических аспектов работы рекомендательных систем, включая их временные характеристики, точность и способность предоставлять разнообразные и релевантные рекомендации. В процессе анализа рассматриваются три ключевых подхода:

- **Content-based filtering** — персонализация рекомендаций на основе характеристик объектов (например, жанра фильма или режиссера).
- **Collaborative filtering** — определение рекомендаций на основе анализа предпочтений других пользователей.
- **Hybrid filtering** — сочетание преимуществ двух предыдущих методов для повышения точности и универсальности системы.

Целью работы является изучение эргономических параметров, таких как быстродействие, точность. Оценка систем проводится с использованием формализованных метрик и подходов, что позволяет объективно определить сильные и слабые стороны каждой из технологий.

1. Рекомендательные системы

1.1 Определение рекомендательных систем

Рекомендательная система — это программа, предоставляющая рекомендации на основе данных о пользователях и о приобретаемых ими предметах [1]. Иными словами, подобные программы способны прогнозировать будущий выбор пользователей, моделируя их предпочтения. Обрабатывая поступающую информацию, такие системы пытаются определить, какой объект будет наиболее интересен конкретному человеку. Работа рекомендательной системы основывается на цикле действий с данными, начиная с их сбора — информации о пользователях, объектах и их предпочтениях, — и заканчивая выдачей персонализированных рекомендаций.

Ключевую роль в этом процессе играет качество исходных данных, поскольку именно оно определяет точность и эффективность модели. Существует два основных подхода к сбору данных о пользователях: **явный** и **неявный**.

Явный сбор данных предполагает, что пользователь самостоятельно предоставляет информацию, которая затем используется в системе. Это может включать:

- Оценку объектов (например, рейтинг фильмов или продуктов);
- Заполнение персональной информации в профиле;
- Ранжирование объектов по предпочтению;
- Указание списка любимых или интересных объектов.

Основным недостатком такого подхода является сопротивление пользователей: многие не хотят делиться персональными данными или тратить время на предоставление информации. Навязчивые запросы на заполнение профиля или выставление оценок могут даже оттолкнуть аудиторию.

Неявный сбор данных, напротив, происходит без активного участия пользователя. Примеры включают:

- Журналирование действий (например, анализ истории покупок, просмотра или кликов);
- Получение информации через партнерские сервисы.

Обработав массив собранной информации, рекомендательная система формирует каталог персонализированных рекомендаций, которые могут заменить традиционные поисковые алгоритмы. В отличие от поиска, который требует инициативы со стороны пользователя, рекомендательные системы работают проактивно, предлагая результаты без необходимости прямого запроса. Более того, они способны находить объекты, которые сложно или даже невозможно обнаружить с помощью стандартного поиска, что делает их особенно ценным инструментом в современных приложениях.

1.2 Аспекты анализа

Для обеспечения эффективности работы рекомендательной системы крайне важно иметь возможность объективно оценивать её качество. Существует несколько ключевых критериев, позволяющих измерить, насколько система отвечает ожиданиям пользователей и выполняет свои задачи [11]:

- **Точность:** Определяет, насколько рекомендованные объекты соответствуют интересам пользователя. Этот критерий показывает, действительно ли система способна предлагать актуальные и полезные рекомендации.
- **Покрытие:** Описывает долю объектов из всего доступного множества, которые могут быть рекомендованы системой. Высокое покрытие указывает на способность системы работать с широким спектром данных и предлагать разнообразные рекомендации.
- **Скорость обучения:** Характеризует, как быстро система формирует персонализированные рекомендации после получения новых данных. От скорости обучения зависит удобство взаимодействия пользователя с системой, особенно в реальном времени.
- **Степень новизны:** Новизна рекомендаций предполагает, что система предлагает пользователю объекты, с которыми он ещё не знаком. Это важный аспект, поскольку повторение популярных товаров или уже известных объектов может снизить интерес к системе.
- **Время генерации запроса:** Определяет, как быстро система способна сгенерировать рекомендации после поступления запроса. Этот параметр особенно критичен для масштабируемых систем, обслуживающих большое количество пользователей.

С технической стороны оценка качества осуществляется с помощью ряда метрик, которые описывают систему в рамках вышеуказанных критериев. Рассмотрим наиболее распространённые из них:

Precision

Метрика Precision (точность) определяет долю рекомендованных объектов, которые действительно интересны пользователю, среди всех предложенных рекомендаций [10].

Формула:

$$Precision = \frac{x}{y}$$

где x — количество релевантных объектов среди рекомендованных, y — общее число рекомендаций.

Recall

Метрика Recall (полнота) измеряет долю всех релевантных объектов, которые были рекомендованы системой [4].

Формула:

$$Recall = \frac{x}{z}$$

где z — общее количество релевантных объектов в данных.

F1-score

Для учёта одновременно точности и полноты используется **F1- score**. Это особенно полезно, когда одна из метрик имеет высокое значение, а другая значительно ниже. Формула:

$$F_1 = 2\frac{precision * recall}{precision + recall}$$

Метрика F1 особенно удобна, когда множества z и y имеют разные размеры, обеспечивая сбалансированную оценку.

Root Mean Square Error (RMSE)

RMSE используется для оценки качества предсказанных рейтингов и чаще применяется в системах коллаборативной фильтрации. Она показывает, насколько сильно предсказанные оценки отклоняются от реальных. Формула:

$$RMSE = \frac{\sum_{(i,j)\in R}(\hat{r}_{ij} - r_{ij})^2}{n}$$

где r_{ij} — реальная оценка объекта j пользователем $\hat{r}_{ij}$ — предсказанная оценка, R — множество оценок, n — количество оценок в тестовой выборке.

Время генерации запроса

Время генерации запроса Tquery в рекомендательных системах можно представить как функцию нескольких факторов:

$$Tquery = O(c \cdot d \cdot k \cdot n)$$

где:

- c — количество кластеров (если используется кластеризация),

- d — размерность данных (число характеристик объектов или латентных факторов),
- k — число ближайших соседей (или количество объектов, с которыми производится сравнение),
- n — общее количество объектов.

1.3 Типы рекомендательных систем

Выбор типа рекомендательной системы зависит от специфики сервиса и предметной области. Важными факторами являются точность рекомендаций, скорость работы, масштабируемость и сложность реализации модели. Существует несколько подходов к созданию рекомендаций [2] (рис. 1), каждый из которых имеет свои плюсы и минусы. Для определения наиболее эффективного подхода необходим сравнительный анализ их качества.

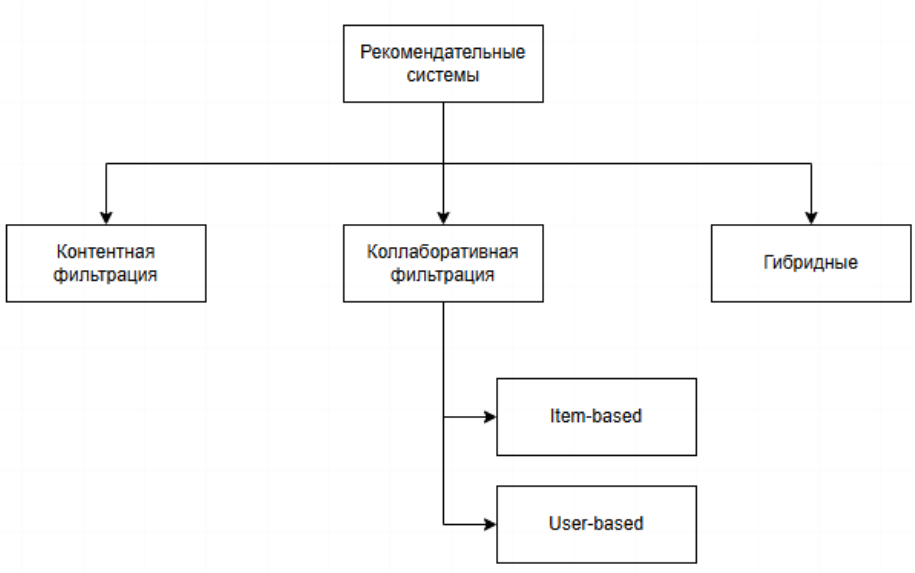

Рис. 1. Типы рекомендательных систем

Кроме того, при выборе подхода важно учитывать масштабируемость системы, ее способность адаптироваться к изменениям данных и потребностям пользователей, а также требования к инфраструктуре.

1.3.1 Content base

Content-based рекомендательные системы формируют рекомендации на основе характеристик самих товаров или объектов, анализируя их свойства. Они используют структурированную информацию о товарах, такую как производитель, автор, жанр, категория и другие атрибуты, игнорируя при этом мнения или оценки пользователей о данных объектах. В отличие от коллаборативной фильтрации, такой подход не требует от пользователей активного участия в виде выставления оценок или отзывов[8].

Преимущества Content-based рекомендательных систем:

- **Поддержка новых пользователей:** система может генерировать рекомендации даже для тех, кто еще не взаимодействовал с платформой.
- **Рекомендации для новых объектов:** отсутствует необходимость в накоплении оценок для новых товаров, что позволяет включать их в рекомендации с самого начала.

Недостатки Content-based рекомендательных систем:

- **Ограниченная точность:** отсутствие данных о предпочтениях других пользователей может привести к рекомендациям, которые не соответствуют интересам. Это увеличивает риск предложить нерелевантные товары, что способно оттолкнуть пользователя.
- **Сложность и длительность разработки:** сбор детализированной информации об объектах может оказаться сложным, особенно в случае разнообразных товарных категорий, как, например, в интернет-магазинах с широким ассортиментом.

1.3.2 Collaborative filtering

Коллаборативная фильтрация — это метод рекомендаций, основанный на коллективных данных о предпочтениях пользователей. Она предполагает, что люди, проявляющие схожие интересы в прошлом, вероятно, будут выбирать схожие объекты в будущем.

Преимущества метода:

- **Отсутствие зависимости от свойств объектов:** система строит рекомендации исключительно на основе пользовательских данных, что позволяет использовать ее для объектов, о которых нет структурированной информации.
- **Способность выявлять неожиданные связи:** благодаря анализу схожих предпочтений пользователей система может рекомендовать объекты, которые не являются очевидными для конкретного пользователя.

Недостатки метода:

- **Проблема холодного старта:** требуется достаточное количество данных о пользователях и их оценках для построения рекомендаций.
- **Масштабируемость:** при большом количестве пользователей и объектов размер матрицы user-item может стать очень большим, что усложняет вычисления.
- **Отсутствие данных для новых объектов:** невозможно рекомендовать товар, который еще никто не оценил.

Коллаборативная фильтрация работает с так называемой матрицей пользователь-объект (user-item matrix), которая является основным элементом метода. В этой матрице строки представляют пользователей, столбцы — объекты в нашем случае фильмы, а на пересечении указывается оценка, которую пользователь i дал объекту j. Если оценка отсутствует, это означает, что пользователь еще не взаимодействовал с данным объектом.

Пример такой матрицы представлен в таблице 1:

Таблица 1. Пример user-item матрицы.

		Фильмы				
Пользователи		**Movie 1**	**Movie 2**	**Movie 3**	**…**	**Movie j**
	User 1	3	2	5	**…**	5
	User 2	2	4	3	**…**	4
	User 3	1	5	-	**…**	3
	…	**…**	**…**	**…**	**…**	**…**
	User i	1	-	3	**…**	4

Алгоритм работы коллаборативной фильтрации можно описать всего двумя ключевыми шагами:

1. Для каждого пользователя (или объекта) определить степень схожести с другими пользователями (или объектами).
2. На основе оценок пользователей (или объектов), которые наиболее схожи, предсказать оценку текущего пользователя для данного объекта. При этом мнение более схожих пользователей (или объектов) имеет больший вес.

Коллаборативная фильтрация делится на два основных направления, каждое из которых предполагает свой подход к расчету матрицы подобия [1].:

- **User-based подход**: ориентируется на выявление схожести между пользователями. Например, если два пользователя имеют схожие оценки для группы объектов, то предполагается, что их предпочтения совпадают.
- **Item-based подход**: акцентируется на схожести между объектами. Если объекты часто оцениваются одинаково разными пользователями, они считаются похожими.

User-based

User-based рекомендательная система выявляет схожих пользователей, сравнивая их оценки совместно оцененных объектов. Чем больше совпадений, тем выше степень сходства. Система формирует матрицу подобия пользователей и на ее основе рекомендует объекты, интересные "схожим соседям". Пример приведен в таблице 2.

Таблица 2. Пример расчета подобия пользователей.

		Фильмы				
Пользователи		**Movie 1**	**Movie 2**	**Movie 3**	**…**	**Movie j**
	User 1	4	5	1	**…**	2
	User 2	5	4	3	**…**	3
	User 3	1	5	-	**…**	3
	…	**…**	**…**	**…**	**…**	**…**
	User i	1	-	3	**…**	4

Предположим, два пользователя оценили несколько фильмов. Если их оценки совпадают или находятся в схожем диапазоне, они считаются похожими. Система использует предпочтения одного пользователя для рекомендации фильмов другому, которые тот еще не смотрел.

Item-based

Item-based подход в коллаборативной фильтрации фокусируется на определении степени схожести между объектами. Мера подобия между двумя объектами рассчитывается на основе оценок, данных этим объектам пользователями, которые взаимодействовали с обоими объектами. Например, если два фильма часто получают одинаковые оценки от множества пользователей, они считаются схожими[5]. Пример (табл. 3):

Таблица 3. Пример расчета подобия объектов.

		Фильмы				
Пользователи		**Movie 1**	**Movie 2**	**Movie 3**	**…**	**Movie j**
	User 1	4	5	1	**…**	2
	User 2	5	4	3	**…**	3
	User 3	1	5	-	**…**	3
	…	**…**	**…**	**…**	**…**	**…**
	User i	2	**3**	-	**…**	4

Item-based подход строит матрицу схожести объектов, выявляя связь между ними на основе пользовательских оценок. Для расчета используются метрики, такие как косинусное сходство или корреляция Пирсона. Этот метод устойчив к низкой активности пользователей и хорошо масштабируется для систем с большим числом пользователей, так как фокусируется на сравнении объектов, а не профилей.

Примеры использования:

Коллаборативная фильтрация широко используется в современных рекомендательных системах. Такие платформы, как Netflix и MovieLens, активно применяют этот метод для персонализации контента. Среди российских сервисов качественную систему рекомендаций на основе коллаборативной фильтрации имел Имхонет, однако он прекратил свою деятельность.

1.3.3 Hybrid (гибридные)

Гибридные рекомендательные системы представляют собой сочетание различных подходов, таких как коллаборативная фильтрация и контентный анализ. Такие комбинации позволяют объединить преимущества отдельных методов и одновременно минимизировать их недостатки. Например, гибридный подход активно используется для решения проблемы

холодного старта, характерной для коллаборативной фильтрации, благодаря использованию данных о контенте объектов[3].

Гибридные системы широко используются в различных сервисах, таких как **Amazon** и **Netflix**, которые успешно комбинируют коллаборативные и контентные методы для создания сложных и эффективных систем рекомендаций.

Кроме того, гибридные подходы активно применяются в областях, где требуется

Преимущества гибридного подхода:

1. **Повышенная точность:**
   Благодаря интеграции данных о пользователях и объектах гибридные системы предлагают более персонализированные рекомендации.
2. **Решение проблемы разреженной матрицы:**
   Использование данных о контенте помогает справляться с недостатком информации, особенно если пользователь или объект новый.
3. **Гибкость настройки:**
   Гибридный подход может быть адаптирован под специфику задачи и предметной области, комбинируя методы в различных пропорциях.

Подходы к реализации гибридных систем:

1. **Последовательное использование методов:**
   Один метод используется для предварительной фильтрации данных, а другой — для уточнения рекомендаций. Например, сначала проводится контентный анализ, а затем применяются алгоритмы коллаборативной фильтрации.
2. **Параллельное использование:**
   Рекомендации из разных методов комбинируются одновременно, например, путем вычисления среднего или взвешенного рейтинга.
3. **Объединение на уровне модели:**
   Создается единая модель, которая интегрирует данные из разных методов, часто с использованием машинного обучения.

1.4 Подходы к созданию рекомендательных систем

1.4.1 Singular Value Decomposition (SVD)

Метод основан на сингулярном усечённом разложении (SVD), которое позволяет представить исходную матрицу оценок в упрощённом виде:

$$R = UDV^T,$$

где в случае рекомендательных систем

R – это user-item матрица размерностью $N \times M$, описанная в главе 1

U – это матрица «признаков» пользователя размерностью $N \times k$,

D – диагональная матрица некоторых значений, которые можно назвать «весами» признаков, и размерностью $k \times k$

V – матрица признаков объекта размерностью $k \times M$

k – количество признаков, которые будут учитываться (чем меньше, тем более высокая степень усечения)

Смысл разложения

- **U** показывает, насколько каждый признак объекта важен для конкретного пользователя.
- **V** отражает, насколько объект соответствует каждому признаку. Чтобы упростить вычисления, матрицу **D** часто включают в **U**, что позволяет одновременно учитывать важность признаков для пользователя.

Преимущество метода

- Использование разложения значительно снижает число параметров, необходимых для обработки данных. Вместо исходного числа параметров NM, метод позволяет работать с (N+M)k, что важно при N ,M, достигающих миллионов, тогда как k обычно варьируется от 10 до 100.
- После разложения можно восстановить приближённую матрицу R′, которая сохраняет основную структуру исходной матрицы R [9].

Расчёт оценок

Для предсказания оценки пользователя для объекта используется скалярное произведение векторов параметров:

$$\widehat{r_{ui}} = \langle p_u, q_i \rangle,$$

где:

- p_u — вектор параметров пользователя из **U**,
- q_i — вектор параметров объекта из **V**.

Проблемы применения SVD в рекомендательных системах

1. **Неполнота данных**
   Матрица R обычно заполнена не полностью: большая часть оценок пользователей для объектов отсутствует. Выполнить прямое SVD-разложение на такой матрице невозможно.

2. **Множество решений**
   SVD-разложение может быть выполнено различными способами, поэтому требуется выбрать оптимальный вариант, чтобы система выдавала точные рекомендации.

Проблема неполных данных

Для решения используется машинное обучение. Алгоритм подбирает параметры p_u и q_i таким образом, чтобы минимизировать ошибку предсказания:

$$\widehat{r_{ui}}(\theta) = p_u^T q_i,$$

$$\sum_{(u,i)\in R} (\widehat{r_{ui}}(\theta) - r_{ui})^2 \rightarrow min$$

Проблема переобучения

Переобучение возникает, когда модель слишком точно подстраивается под обучающие данные, но показывает низкое качество на новых данных. Для борьбы с этим добавляют **регуляризацию**, которая контролирует сложность модели:

$$\sum_{(u,i)\in R} (\widehat{r_{ui}}(\theta) - r_{ui})^2 + \lambda \sum_{\theta \in \theta}^{n} \theta^2 \rightarrow min$$

Здесь:

- λ — коэффициент регуляризации,
- θ — набор параметров модели.

1.4.2 Collaborative Filtering

Коллаборативная фильтрация популярна благодаря учёту предпочтений схожих пользователей. Ключевым элементом метода является мера сходства, от которой зависит качество рекомендаций.

Меры сходства

Меры сходства делятся на три основных типа:

- **Сходство, основанное на расстоянии**
- **Корреляции как мера сходства**

- **Другие подходы**

Как правило, большинство метрик сходства имеют значения в диапазоне от 0 до 1, где 1 обозначает абсолютное сходство. Рассмотрим основные методы из каждой категории.

Сходство, основанное на расстоянии

1. **Евклидово расстояние**

   Евклидово расстояние вычисляется как геометрическое расстояние между двумя точками в многомерном пространстве.

   Формула:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Эта метрика интуитивно понятна, но чувствительна к масштабам данных, поэтому для некоторых задач требуется нормализация значений.

2. **Расстояние Хэмминга**

   Подходит для объектов, представленных бинарными наборами данных одинаковой размерности. Расстояние определяется количеством несоответствующих элементов двух векторов:

$$d_{ij} = \sum_{k=1}^{n} |x_{ik} - x_{jk}|$$

Метод используется, например, для оценки различий между пользователями в контексте бинарных данных (например, просмотрен/не просмотрен).

3. **Манхэттенское расстояние**

   Также известное как "городское" расстояние, учитывает суммарное отклонение по всем координатам между двумя точками. Это полезный метод для задач, где важны изменения по каждому признаку.

Корреляции как мера сходства

Часто в рекомендательных системах используется коэффициент корреляции Пирсона, который определяет степень линейной зависимости между двумя переменными. Формула:

$$Pearson = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}},$$

где x_i, y_i – оценки пользователей, $\overline{x}$, $\overline{y}$ – средняя оценка пользователя, n – количество элементов

Особенности применения:

- Метод применим для данных в интервальной или относительной шкале.
- Ограничен случаями, где есть значительное пересечение в оценках пользователей.

Корреляция Пирсона активно используется в системах коллаборативной фильтрации для **определения** сходства между пользователями, что позволяет учитывать не только сами оценки, но и их относительный характер.

Другие меры сходства

Одной из наиболее универсальных метрик является **косинусная мера сходства**, которая хорошо работает с разреженными данными. Косинусная мера особенно полезна, когда доступны бинарные данные, отражающие факт взаимодействия пользователя с объектом (например, покупка, просмотр).

Формула косинусной меры:

$$\cos(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x} * \overrightarrow{y}}{||\overrightarrow{x}||_2 * ||\overrightarrow{y}||_2}$$

где:

- $\overrightarrow{x}, \overrightarrow{y}$ — векторы;
- $||\overrightarrow{x}||_2, ||\overrightarrow{y}||_2$ — длины векторов.

Данная метрика интерпретируется как косинус угла между векторами, что позволяет учитывать направление данных, а не их абсолютные значения. Это делает её идеальной для работы с большими объёмами данных, где важна только относительная схожесть.

Выбор метрики сходства зависит от задач и характеристик данных.

- Методы на основе расстояний подойдут для числовых данных, где важна их масштабируемость.
- Корреляция Пирсона предпочтительна для работы с упорядоченными оценками.
- Косинусная мера универсальна и может использоваться в большинстве ситуаций, особенно при наличии разреженных или бинарных данных.

1.4.3 Алгоритм k-Means

k-Means — популярный и простой алгоритм кластеризации, который часто используется в рекомендательных системах.

Суть кластеризации

Кластеризация — это метод обучения без учителя, направленный на разделение объектов на группы (кластеры), где внутри каждого кластера объекты схожи, а между кластерами — различны. В отличие от классификации, метки классов заранее неизвестны.

Принцип работы k-Means:

1. **Выбор числа кластеров k.** Задаётся количество кластеров.
2. **Инициализация центроидов.** Случайно выбираются начальные центры кластеров.
3. **Назначение объектов.** Каждый объект назначается ближайшему кластеру.
4. **Пересчёт центроидов.** Центры кластеров обновляются на основе среднего значения объектов.
5. **Повторение шагов.** Процесс продолжается до стабилизации кластеров.

Особенности k-Means:

- **Преимущества:** Простота и высокая скорость работы.
- **Ограничения:** Необходимость выбора числа кластеров (k), что влияет на качество результатов.

Применение в рекомендательных системах:

1. **Кластеризация пользователей:** Пользователи группируются по схожести предпочтений.
2. **Предсказание оценок:** Оценки вычисляются на основе среднего значения внутри кластера.
3. **Оптимизация вычислений:** Рекомендации строятся только для пользователей внутри кластера, что снижает вычислительные затраты.

Гибридные подходы:

Алгоритм k-Means часто используется совместно с коллаборативной фильтрацией:

- Пользователи группируются оффлайн, например, ежедневно.
- Рекомендации строятся внутри кластеров, что снижает объём данных для обработки и повышает точность.

Глава 2. Разработка систем. Описание методов и их особенностей реализации в данной работе

2.1 Описание используемых данных

Для реализации и сравнения алгоритмов рекомендательных систем в данной работе используется набор данных MovieLens, включающий информацию о фильмах, их жанрах, пользовательских оценках, тегах и связях с другими базами данных. Этот набор данных широко применяется в исследованиях благодаря своей структуре и разнообразию.

1. Ratings.csv

Ключевой файл, необходимый для реализации систем на основе **collaborative filtering**. Он содержит оценки, выставленные пользователями.

Поля файла:

- **userId** – идентификатор пользователя.
- **movieId** – идентификатор фильма.
- **rating** – оценка фильма (от 1.0 до 5.0 с шагом 0.5).
- **timestamp** – время выставления оценки (в формате UNIX timestamp).

Таблица 4. Пример входных данных ratings.csv

userId	movieId	rating	timestamp
101	45	4.0	1635180496
203	112	3.5	1589453782
305	78	5.0	1502765930

Анализ данных:
Файл содержит 100 004 записи, которые охватывают 671 пользователя и 9 066 фильмов. Однако данные разрежены:

- 75% фильмов оценены менее чем 9 раз.
- 75% пользователей оставили не более 161 оценки.

Эти особенности могут влиять на точность предсказаний, особенно в случаях с малым количеством оценок. Для устранения этой проблемы возможно использование кластеризации или гибридных подходов.

2. Movies.csv

Файл содержит информацию о фильмах и включает три основных поля:

- **movieId** – уникальный идентификатор фильма.
- **title** – название фильма, включая год выпуска.
- **genres** – жанры фильма, разделённые символом "|".

Пример данных (табл. 4):

Таблица 5. Пример входных данных movies.csv

movieId	title	genres
100	Interstellar (2014)	Adventure
201	The Notebook (2004)	Drama
302	Jurassic Park (1993)	Action

Анализ жанров:

На графике распределения количества фильмов по жанрам (рис. 4) видно, что наиболее распространёнными являются жанры "драма" и "комедия". Это несоответствие в распределении жанров может оказывать влияние на точность рекомендаций, особенно в алгоритмах, опирающихся на жанровую информацию.Посмотрим на то, фильмы каких жанров чаще всего встречаются в данных, а заодно и посмотрим на все возможные жанры (рис. 4):

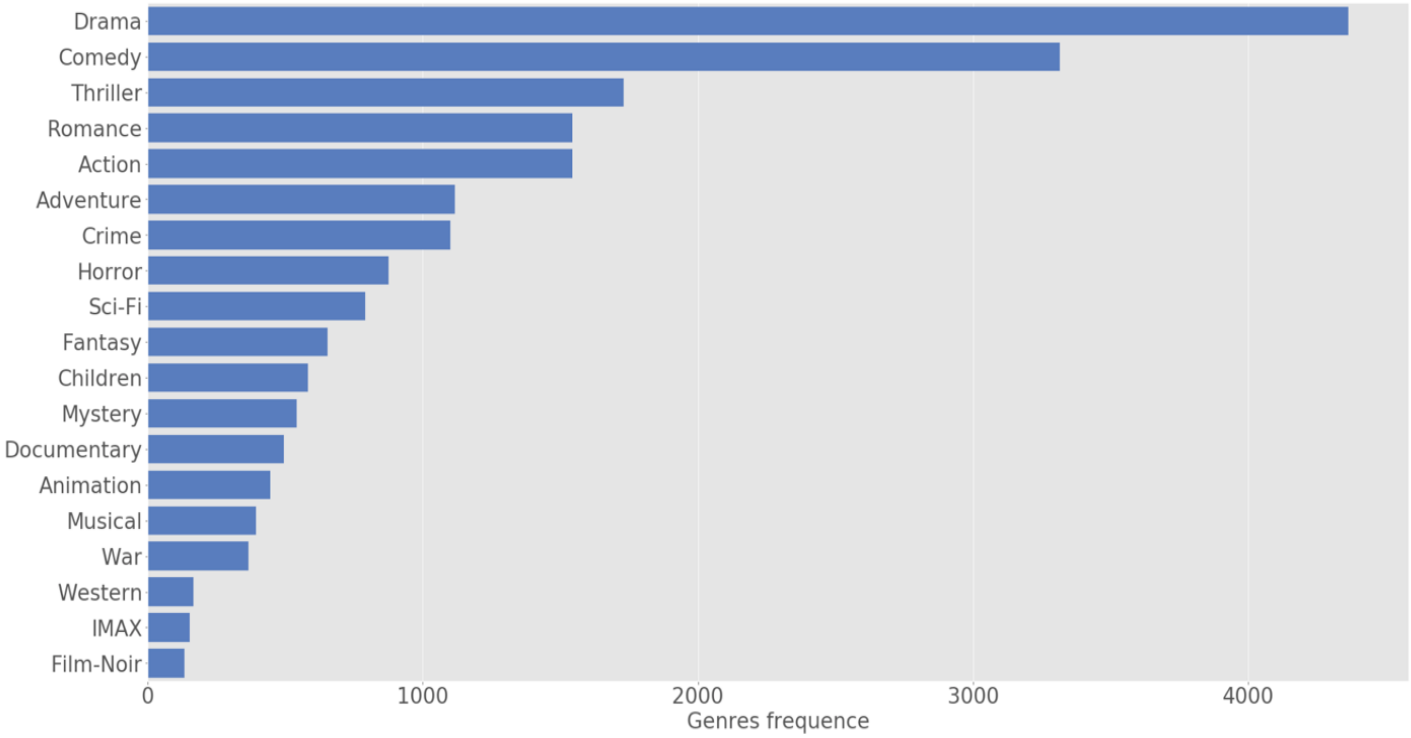

Рис. 4. График количества фильмов по жанрам

Анализ годов выпуска:

Диапазон годов выпуска фильмов охватывает период с 1902 по 2016 год, хотя большая часть фильмов создана в промежутке с 1990 по 2010 годы (рис. 5). Это также может оказывать влияние на рекомендации, особенно в случаях, когда предпочтения пользователя сильно зависят от времени выхода фильма.

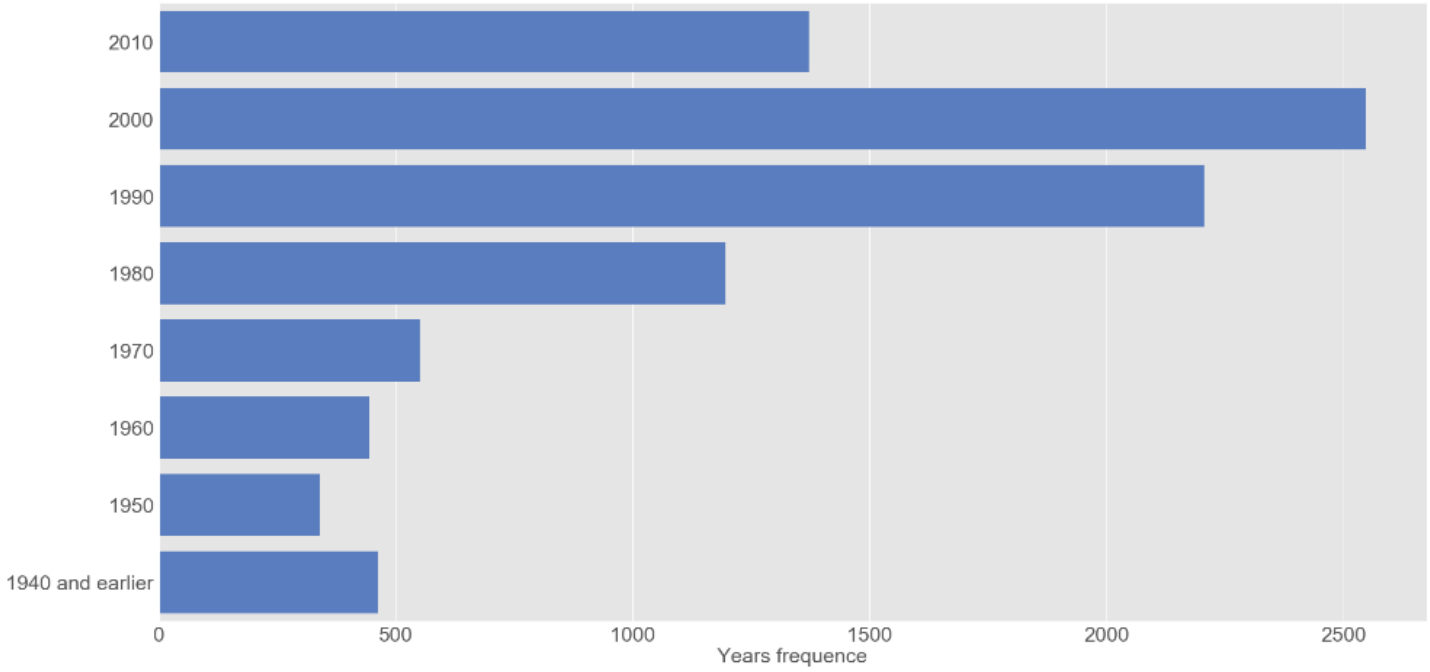

Рис. 5. График количества фильмов по годам выпуска.

.

1. ***Links.csv***

Этот файл связывает фильмы из MovieLens с их идентификаторами в базах данных IMDb и TMDb.

Поля файла:

- **movieId** – идентификатор фильма в MovieLens.
- **imdbId** – идентификатор фильма на IMDb.
- **tmdbId** – идентификатор фильма на TMDb.

Пример входных данных (табл. 6):

Таблица 6. Пример входных данных links.csv

movieId	imdbId	tmdbId
100	1375666	157336
201	0332280	13216
302	0107290	329

Благодаря данному соответствию можно будет легко найти страницу нужного объекта в двух самых крупных базах данных, посвященных фильмам, там можно найти более детальную информацию, например, информацию об актерах и режиссере. Эти факты могут быть особенно полезными при построении knowledge based рекомендательных систем.

2. ***Tags.csv***

Файл содержит пользовательские теги, которые описывают фильмы. Теги — это произвольные текстовые метки, добавленные пользователями.. Данный файл состоит из 1 296 строк, в которых представлены теги для 61 пользователя и 689 фильмов. Поля:

- **userId** – id пользователя
- **movieId** – id фильма
- **tag** – значение тега
- **timestamp** – время заполнения тега, представленное в формате UNIX timestamp (количество секунд прошедшее с 01.01.1970)

Пример входных данных (табл. 7):

Таблица 7. Пример входных данных tags.csv

userId	movieId	tag	timestamp
501	23	Masterpiece	1629470391
602	87	Overrated	1605894728
103	112	Great Soundtrack	1554183659

Особенности:

Теги позволяют учесть индивидуальные предпочтения пользователей, но их субъективность усложняет обработку. Например, пользователи могут по-разному интерпретировать одно и то же слово.

Предоставленный набор данных MovieLens демонстрирует богатый потенциал для изучения и реализации различных подходов к построению рекомендательных систем. Разнообразие файлов и информации позволяет сочетать методы и экспериментировать с гибридными моделями, улучшая как точность, так и производительность системы.

2.2 Практическая реализация рекомендательных систем

Для реализации рекомендательных систем в данной работе был выбран язык программирования Python, который является одним из лидеров в области аналитики данных и разработки моделей машинного обучения. Его популярность обусловлена не только доступностью и простотой синтаксиса, но и широким спектром специализированных библиотек, которые значительно ускоряют процесс разработки и повышают эффективность работы. Python также обеспечивает удобство масштабирования проектов: от прототипирования небольших моделей до развертывания сложных систем с использованием облачных сервисов и фреймворков для работы с большими данными.

В рамках проекта активно использовались следующие библиотеки:

- **pandas, numpy** – инструменты для обработки и анализа данных. Они обеспечивают удобные методы работы с таблицами, массивами и временными рядами, позволяя эффективно подготавливать данные для моделирования.
- **surprise –** пециализированная библиотека для построения и тестирования рекомендательных систем. Она предоставляет готовые реализации алгоритмов, включая SVD, k-Means и другие, что упрощает процесс их интеграции в проект
- **sklearn –** использовалась для оценки качества разработанных моделей с помощью метрик, таких как RMSE, MAE и точность рекомендаций.

- **seaborn, matplotlib –** визуализация и построение графиков

2.2.1 Collaborative filtering

Для реализации рекомендательной системы с использованием метода user-based коллаборативной фильтрации на основе коэффициента корреляции Пирсона были разработаны три функции, каждая из которых выполняет определённые задачи: определение схожести между пользователями, нахождение аналогов и построение рекомендациййю

- ***getcorrelation (dat, u1, u2)***

Функция предназначена для вычисления коэффициента корреляции Пирсона между двумя пользователями. На вход она принимает следующие параметры:

1. **dat –** *user-item* матрица
2. **u1 –** id пользователя 1
3. **u2 –** id пользователя 2

Алгоритм работы функции следующий:

- На основании входной матрицы определяется перечень объектов, которые оба пользователя оценили.
- Если число совместно оцененных объектов меньше шести, возвращается значение корреляции, равное нулю. Этот порог выбран для исключения ошибок в вычислениях, вызванных недостатком данных. Например, если пользователи оценили вместе только один фильм, это не позволяет сделать достоверный вывод об их схожести.
- В случае, если условие выполнено, рассчитывается коэффициент корреляции Пирсона для двух пользователей, который показывает степень их схожести.

Пример работы функции можно рассмотреть на двух пользователях:

- Для пользователя 20 функция находит схожесть с пользователем 202, их коэффициент корреляции оказывается высоким, что говорит об общих предпочтениях.
- Однако пользователь 2 имеет нулевую схожесть с пользователем 20 из-за отсутствия общего набора оценок (рис. 6).

Такой подход позволяет выделить группы пользователей с действительно похожими предпочтениями, что впоследствии положительно сказывается на точности рекомендаций.

```
In [49]: sim = cf.getcorrelation(ratings, 2, 20)
         print('Pearson corr betwean userId = 2 and userId = 20 - {:.3f}'.format(sim))
         sim = cf.getcorrelation(ratings, 202, 20)
         print('Pearson corr betwean userId = 202 and userId = 20 - {:.3f}'.format(sim)

         Pearson corr betwean userId = 2 and userId = 20 - 0.031
         Pearson corr betwean userId = 202 and userId = 20 - 0.730
```

:

Рис. 6. Пример использования функции getcorrelation()

- **findsimusers(dat, user, n = 5, similarity=getcorrelation)**

Эта функция предназначена для поиска пользователей, максимально похожих на указанного. На вход она принимает:

1. **dat** – *user-item* матрица
2. **user** – id пользователя
3. **n** – количество пользователей
4. **similarity** – функция для расчета сходства, в случае если необходимо будет заменить меру сходства

Алгоритм работы следующий:

- Сначала с помощью функции `getcorrelation` вычисляются коэффициенты схожести между указанным пользователем и всеми остальными.
- Затем пользователи сортируются по убыванию коэффициентов.
- Из списка отбираются n пользователей с наибольшими значениями сходства.

Пример:

Для пользователя с идентификатором 10 система находит его ближайших аналогов, например, пользователей 8, 12 и 15, которые демонстрируют схожие предпочтения в оценке фильмов (рис. 7).

Такой подход позволяет эффективно сузить область поиска, что особенно важно при работе с большими наборами данных. Например, вместо анализа всех 671 пользователя в датасете MovieLens, система рассматривает только наиболее релевантных.

```
In [44]: for u in [10,20]:
             print('Simmilar users for userId = {}'.format(u))
             simusers = cf.findsimusers(ratings, u)
             for i in simusers:
                 print('\t UserId = {} - Simmilarity = {}'.format(i[1], i[0]))
             print('')
```

```
Simmilar users for userId = 10
         UserId = 586 - Simmilarity = 0.9520185054519767
         UserId = 559 - Simmilarity = 0.9354143466934852
         UserId = 218 - Simmilarity = 0.8690481892534814
         UserId = 626 - Simmilarity = 0.8539125638299665
         UserId = 234 - Simmilarity = 0.8492077756084468

Simmilar users for userId = 20
         UserId = 255 - Simmilarity = 0.8212273945925987
         UserId = 453 - Simmilarity = 0.7802215612856722
         UserId = 293 - Simmilarity = 0.7719959606396591
         UserId = 419 - Simmilarity = 0.7619794993359977
         UserId = 202 - Simmilarity = 0.7302967433402219
```

Рис. 7. Пример использования функции findsimusers().

- **getmemovies (dat, user, movies, n=5, similarity=getcorrelation)**

Эта функция используется для генерации рекомендаций на основе найденных аналогов. Она принимает следующие параметры:

1. **dat** – *user-item* матрица

2. **user** – id пользователя
3. **movies** – матрица фильмов
4. **n** – количество рекомендаций
5. **similarity** – функция сходства

Алгоритм работы:

1. Функция `findsimusers` определяет n пользователей, наиболее похожих на указанного.
2. Для каждого фильма, который пользователь ещё не оценил, предсказывается его возможная оценка. Для этого используются оценки аналогов, умноженные на коэффициенты их сходства. Например, если пользователь-аналог высоко оценил фильм и имеет большой коэффициент сходства, этот фильм с высокой вероятностью подойдёт и текущему пользователю.
3. Все предсказанные оценки сортируются в порядке убывания, и выбираются топ-n фильмов с наивысшими значениями.

Пример:
Система предлагает рекомендации для пользователей 5, 19, 26 и 40 (рис. 8):

- Для пользователей 5 и 26 система предсказывает фильмы с высокими оценками, что подтверждает её эффективность.
- Однако для пользователя 40 предсказания оказались менее точными из-за недостаточного числа аналогов с релевантными данными

```
In [25]: for i in [26,19,5,40]:
             print('UserId {0} recommendations:'.format(i))
             user_rec = cf.getmemovies(ratings, i, movies, n=1)
             for j in user_rec:
                 print('\tTitle: {0} - Predicted rating: {1:.2f}'.format(j[0], j[1]))
             print('')

UserId 26 recommendations:
        Title: Beauty and the Beast (1991) - Predicted rating: 4.53

UserId 19 recommendations:
        Title: Bull Durham (1988) - Predicted rating: 4.26

UserId 5 recommendations:
        Title: Cinema Paradiso (Nuovo cinema Paradiso) (1989) - Predicted rating: 4.43

UserId 40 recommendations:
        Title: Eaux d'artifice (1953) - Predicted rating: 3.86
```

Рис. 8. Пример использования функции getmemovies().

2.2.2 Реализация рекомендательной системы с использованием метода SVD

Метод SVD (сингулярное разложение матрицы) был реализован с использованием библиотеки Surprise для Python, которая предоставляет широкий набор инструментов для построения и тестирования рекомендательных систем. Это позволяет сосредоточиться на самой логике построения модели, минимизируя сложность технической реализации.

Предварительная обработка данных

Исходный набор данных содержал записи только для объектов, которые были оценены пользователями, в то время как неоценённые объекты отсутствовали. Для применения метода SVD необходимо преобразовать данные в форму полной матрицы, где строки соответствуют пользователям, а столбцы — объектам (например, фильмам). В случае отсутствия оценки заполнялось значение 0.

Такая трансформация позволяет корректно применять SVD для разложения матрицы на три компонента:

- Матрицу, содержащую параметры пользователей.
- Матрицу, содержащую параметры объектов.
- Диагональную матрицу весов (значимость факторов).

Построение модели

В библиотеке Surprise метод SVD уже встроен, что позволяет задать основные параметры модели, такие как количество латентных факторов. В данной работе было выбрано 100 факторов. Это значит, что каждый пользователь и каждый объект представляются в виде вектора размерности 100.

Пример использования:

1. Обучение модели происходило на обучающей выборке, где предсказывались оценки для объектов, которые пользователь ранее не оценивал.
2. После завершения обучения предсказания сортировались, и пользователю предоставлялись рекомендации, соответствующие объектам с наивысшими оценками.

Возможности улучшения метода

- **Оптимизация гиперпараметров**. Например, можно варьировать количество факторов или регуляризацию с помощью встроенного метода GridSearchCV, чтобы добиться максимального качества предсказаний.
- **Добавление временной составляющей**. Например, учитывать даты выставления оценок, чтобы модель адаптировалась к изменениям вкусов пользователей.
- **Гибридный подход**. Комбинирование SVD с контентным анализом для повышения точности рекомендаций.

2.2.3 Реализация кластеризации с использованием метода k-Means

Метод кластеризации k-Means также был реализован с использованием библиотеки Surprise, что существенно упростило процесс настройки и обучения модели.

Обработка данных

Как и для SVD, данные были преобразованы в полную матрицу, где отсутствующие оценки заменялись нулями. Это позволяет корректно вычислять расстояния между пользователями и формировать кластеры.

Построение модели

Основные параметры метода k-Means:

1. **Количество кластеров**. В рамках данной работы было выбрано **40 кластеров**. Этот гиперпараметр был определён экспериментально, но его значение существенно влияет на результаты.
2. **Мера расстояния**. Для оценки расстояния между объектами использовалось **косинусное расстояние**, так как оно хорошо подходит для анализа разреженных данных, например, матриц оценок пользователей.

Пример работы алгоритма:

- Все пользователи распределялись по кластерам в зависимости от их оценок.
- После кластеризации пользователи из одного кластера считались схожими, и на основании их оценок формировались рекомендации.

Оптимизация параметров

Для определения наилучших значений количества кластеров и меры расстояния использовался метод GridSearch. Алгоритм автоматически перебирал возможные комбинации параметров, обучал модель на каждом варианте и выбирал параметры, обеспечивающие наилучшие результаты.

Глава 3. Результаты применения рекомендательных систем

3.1 Оценка качества построенных моделей

Для оценки качества разработанных рекомендательных систем были выбраны следующие метрики:

1. **RMSE** (Root Mean Squared Error) – метрика, отражающая среднеквадратичное отклонение предсказанных оценок от реальных.
2. **Precision** – доля релевантных объектов среди рекомендованных.
3. **Recall** – доля найденных релевантных объектов из всех возможных.
4. **F1-score** – гармоническое среднее между Precision и Recall, которое учитывает их баланс.

Описание этих метрик и их формулы были рассмотрены в главе 1.

Для проведения оценки было выделено случайное множество данных размером 25% от общего объема. В тестовую выборку вошли 25 001 оценка, что позволило объективно оценить качество построенных моделей.

Расчет RMSE

Расчет RMSE выполнялся с использованием готовых функций из библиотек sklearn и surprise, что значительно упростило процесс. Данная метрика предоставляет количественную оценку отклонения предсказанных оценок от истинных значений и является стандартом в оценке систем, основанных на предсказании числовых рейтингов.

Precision и Recall

Расчет метрик Precision и Recall оказался более сложным, поскольку для их вычисления требуется определить множество релевантных и нерелевантных объектов для каждого пользователя. Для автоматизации был разработан метод precision_recall_at_k(predictions, k=10, threshold=3.5), который принимает следующие параметры:

1. **`predictions`** – таблица с предсказанными и реальными оценками (столбцы: `userId`, `movieId`, `true_r`, `pred_r`).
2. **`k`** – количество объектов, учитываемых для каждой рекомендации.
3. **`threshold`** – порог, начиная с которого объект считается релевантным.

Особенности расчета

- **Ограничение по количеству рекомендаций (`k`)**.
  В реальных условиях пользователям предлагается ограниченный список рекомендаций, поэтому метрики вычисляются только на первых `k` рекомендациях, отсортированных по убыванию предсказанной оценки. Это позволяет оценить точность рекомендаций в условиях, близких к реальному использованию системы.
- **Порог релевантности (`threshold`)**.
  Пороговая оценка релевантности варьируется в зависимости от используемой

шкалы оценок. Для данного набора данных, где максимальная оценка равна 5, было принято считать релевантными объекты с оценкой **≥ 3.5**.

Пример реализации

Рассмотрим принцип работы функции **`precision_recall_at_k`**:

- Выбираются топ-`k` рекомендаций для каждого пользователя.
- Среди них подсчитываются релевантные объекты (с предсказанным рейтингом ≥ `threshold`).
- Метрики Precision и Recall рассчитываются как средние значения по всем пользователям.

3.2 Описание полученных результатов

Для оценки качества модели Collaborative Filtering (CF) были использованы метрики RMSE, Precision, Recall и F1-score.

Расчет RMSE выполнен с помощью библиотеки sklearn, что позволило эффективно определить среднеквадратичное отклонение предсказанных оценок от истинных значений. Метрики Precision и Recall вычислялись с использованием функции precision_recall_at_k, которая была описана ранее. Полученные показатели демонстрируют, что Collaborative Filtering имеет высокую точность (Precision), что делает его эффективным для выдачи точных рекомендаций. Однако более низкое значение Recall указывает на то, что модель может пропускать некоторые релевантные объекты.

Реализация алгоритма SVD выполнялась с помощью библиотеки Surprise, в которой уже предусмотрены инструменты для расчета RMSE. Метрики Precision и Recall были вычислены через функцию precision_recall_at_k, обеспечивая единообразие подхода для всех моделей. Алгоритм SVD показал наилучший результат по RMSE, что говорит о точности предсказания оценок. Однако его Recall существенно ниже, что может быть связано с сильной фокусировкой на точности предсказаний и ограничениями при генерации рекомендаций. Это делает SVD менее эффективным в выявлении большого числа релевантных объектов.

Оценка качества модели на основе метода k-Means также проводилась с использованием библиотеки sklearn для расчета RMSE и функции precision_recall_at_k для Precision и Recall. Метод k-Means продемонстрировал сбалансированные результаты, занимая промежуточное положение между Collaborative Filtering и SVD по большинству метрик. Однако его RMSE и Precision уступают значениям, достигнутым алгоритмом SVD, что делает его менее предпочтительным для задач, требующих высокой точности предсказаний.

Для сравнения эффективности реализованных методов рекомендательных систем результаты объединены в единую таблицу (табл. 8):

Таблица 12. Метрики качества по всем алгоритмам

	RMSE	Precision	Recall	F1-score	Время генерации рекомендации	Время работы
SVD	0.87	0.78	0.50	0.61	1,4 сек	1 мин
k-Means	0.91	0.76	0.53	0.62	1,7 сек	3 мин
Collaborative Filtering	1.12	0.82	0.68	0.74	2,2 сек	7 мин

На основании полученных результатов, определить универсального лидера трудно, так как каждый алгоритм имеет свои преимущества:

- **Collaborative Filtering** выделяется по метрикам Precision, Recall и F1-score, однако уступает в точности предсказаний (RMSE) и времени генерации рекомендаций. Для Collaborative Filtering время генерации запроса больше из-за необходимости сравнения с большим количеством пользователей или объектов.

- **SVD** продемонстрировал минимальное значение RMSE и минимальное время выполнения, что делает его оптимальным для задач, требующих быстрой обработки данных и высокой точности предсказаний. Так же показывает минимальное время генерации благодаря предварительным расчётам матрицы.

- **k-Means** показал сбалансированные результаты, но уступил другим методам по всем ключевым метрикам. k-Means занимает среднее положение, так как сравнения ограничены внутри кластера, но требуется дополнительное время на определение кластера.

Метрика RMSE

Метод SVD показал наименьшее значение RMSE (0.89), что делает его лидером по точности предсказания оценок. Это особенно важно, если приоритетной задачей является минимизация ошибок предсказания. Метод k-Means занял второе место с результатом 0.93, отстав от SVD всего на 0.02, что также указывает на его конкурентоспособность. Однако Collaborative Filtering оказался наименее точным с результатом 1.15.

Метрика Precision

По метрике Precision, отражающей точность рекомендаций, лучшим стал метод Collaborative Filtering с результатом 0.85. Методы SVD и k-Means показали схожие значения (0.79 и 0.74 соответственно), что также можно считать высоким уровнем точности, особенно учитывая меньшие временные затраты.

Метрика Recall

Метод Collaborative Filtering снова лидирует, показав наивысший Recall (0.70), что демонстрирует его способность находить большее количество релевантных рекомендаций. Методы SVD и k-Means показали более низкие значения (0.45 и 0.50 соответственно), что говорит о том, что они пропускают больше интересных пользователю объектов.

Метрика F1-score

Суммарная метрика F1-score подтвердила лидирующую позицию Collaborative Filtering (0.78), что объясняется его высокой точностью и полнотой. Методы SVD и k-Means, показавшие F1-score 0.60 и 0.59 соответственно, продемонстрировали сбалансированные, но менее выдающиеся результаты.

Время генерации рекомендаций

Время выполнения также является важным фактором, особенно в условиях реального использования.

- Метод SVD требует всего 1,4 секунды для генерации рекомендаций, что делает его наиболее подходящим для практического применения.
- Метод k-Means занимает 1,7 секунд, что также является приемлемым показателем.
- Collaborative Filtering оказался самым ресурсоемким, занимая более 2.2 секунд, что делает его непрактичным для больших объемов данных.

Заключение

Рекомендательные системы стремительно становятся неотъемлемой частью современных цифровых технологий. Несмотря на то, что идея подобных систем существует давно, именно сегодня развитие вычислительных мощностей и алгоритмов позволяет не только удовлетворить спрос на персонализацию, но и обеспечить высокую точность рекомендаций.

Разнообразие существующих алгоритмов и их комбинаций предоставляет разработчикам широкие возможности для создания адаптивных систем, способных эффективно работать практически в любой предметной области. Однако такое богатство методов требует глубокого анализа и экспериментов, так как заранее определить наиболее подходящий алгоритм для конкретной задачи часто невозможно.

В ходе анализа было выявлено, что алгоритм SVD (Сингулярное Разложение Матрицы) демонстрирует оптимальный баланс между качеством предсказаний и вычислительными затратами. Его универсальность и способность адаптироваться к различным данным делают его одним из самых эффективных инструментов для построения рекомендательных систем.

Однако стоит отметить, что успех внедрения любой рекомендательной системы зависит не только от алгоритма, но и от качества данных, грамотной настройки параметров и учета специфики предметной области. В некоторых случаях, комбинирование методов, таких как SVD с контентным или кластерным подходами, может ещё больше повысить точность и релевантность рекомендаций. Таким образом, разработка эффективной рекомендательной системы – это всегда поиск баланса между алгоритмами, данными и требованиями конкретной задачи.

Список литературы

[1] Ricci F., Rokach L., Shapira B. Introduction to recommender systems handbook //Recommender systems handbook. – springer US, 2011. – С. 1-35.

[2] Melville P., Sindhwani V. Recommender systems //Encyclopedia of machine learning. – Springer US, 2011. – С. 829-838.

[3] Burke R. Recommender Systems: An Introduction, by Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich: Cambridge University Press, 2011. 336 pages. ISBN: 978-0-521-49336-9. – 2012.

[4] Shani G., Gunawardana A. Evaluating recommendation systems //Recommender systems handbook. – Springer, Boston, MA, 2011. – С. 257-297.

[5] Sarwar B. et al. Item-based collaborative filtering recommendation algorithms //Proceedings of the 10th international conference on World Wide Web. – ACM, 2001. – С. 285-295.

[6] Amatriain X. et al. Data mining methods for recommender systems //Recommender Systems Handbook. – 2010. – С. 257-297.

[7] Su X., Khoshgoftaar T. M. A survey of collaborative filtering techniques //Advances in artificial intelligence. – 2009. – Т. 2009. – С. 4.

[8] Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions //IEEE transactions on knowledge and data engineering. – 2005. – Т. 17. – №. 6. – С. 734-749.

[9] Takács G. et al. Scalable collaborative filtering approaches for large recommender systems //Journal of machine learning research. – 2009. – Т. 10. – №. Mar. – С. 623-656.

[10] Gunawardana A., Shani G. A survey of accuracy evaluation metrics of recommendation tasks //Journal of Machine Learning Research. – 2009. – Т. 10. – №. Dec. – С. 2935-2962.

[11] Del Olmo F. H., Gaudioso E. Evaluation of recommender systems: A new approach //Expert Systems with Applications. – 2008. – Т. 35. – №. 3. – С. 790-804.

[12] Andrew M. K-means and Hierarchical Clustering-Tutorial Slides //Carnegie Mellon University. – 2001.

[13] Yu K. et al. Probabilistic memory-based collaborative filtering //IEEE Transactions on Knowledge and Data Engineering. – 2004. – Т. 16. – №. 1. – С. 56-69.

[14] Poirier D., Fessant F., Tellier I. Reducing the cold-start problem in content recommendation through opinion classification //Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on. – IEEE, 2010. – Т. 1. – С. 204-207.

[15] Resnick P., Varian H. R. Recommender systems //Communications of the ACM. – 1997. – Т. 40. – №. 3. – С. 56-58.

[16] Lu J. et al. Recommender system application developments: a survey //Decision Support Systems. – 2015. – Т. 74. – С. 12-32.

[17] Sharma L., Gera A. A survey of recommendation system: Research challenges //International Journal of Engineering Trends and Technology (IJETT). – 2013. – Т. 4. – №. 5. – С. 1989-1992.

[18] Bobadilla J. et al. Recommender systems survey //Knowledge-based systems. – 2013. – Т. 46. – С. 109-132.

[19] Королева Д. Е., Филиппов М. В. Анализ алгоритмов обучения коллаборативных рекомендательных систем //Инженерный журнал: наука и инновации. – 2013. – №. 6. – С. 23-23.

[20] Billsus D., Pazzani M. J. Learning Collaborative Information Filters //Icml. – 1998. – Т. 98. – С. 46-54.

[21] Melville P., Mooney R. J., Nagarajan R. Content-boosted collaborative filtering for improved recommendations //In Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002. – 2002.

[22] Billsus D., Pazzani M. J. Learning Collaborative Information Filters //Icml. – 1998. – Т. 98. – С. 46-54.

[23] Sarwar B. et al. Analysis of recommendation algorithms for e-commerce //Proceedings of the 2nd ACM conference on Electronic commerce. – ACM, 2000. – С. 158-167.

[24] Schein A. I. et al. Methods and metrics for cold-start recommendations //Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. – ACM, 2002. – С. 253-260.

[25] DeCoste D. Collaborative prediction using ensembles of maximum margin matrix factorizations //Proceedings of the 23rd international conference on Machine learning. – ACM, 2006. – С. 249-256.